December 1990
Vol. 1 No. 8

COBB

Inside DOS™

Tips & techniques for MS-DOS and PC-DOS

Protecting a file against changes using the ATTRIB command

If your computer has a hard drive, you'll likely keep dozens (or even hundreds) of files stored on that drive. Some of those files will change on a regular basis, such as the ones that store your personal telephone directory and your company's organizational chart. On the other hand, some of the files on your hard disk will probably never change or be removed, such as an application program file or last year's income statement.

Because you'll want to prevent changes or modifications to some of the files on your hard disk, you might want to take advantage of the ATTRIB command's ability to protect a file against unwanted changes. In this article, we'll show you how to use the ATTRIB command for this purpose, and mention a few situations in which file protection is useful.

Displaying a file's read-only status

Unfortunately, you can't determine whether a file is read-only by using the DIR command. Instead, you'll need to use the ATTRIB command to bring up this information. To view the status of the read-only attributes for all the files in the current directory, issue the command

```
C:\DATA>attrib *.*
```

Immediately, DOS will display a listing like this:

```
A       C:\DATA\BUDGET90.WK1
A    R  C:\DATA\INCOME89.WK1
A    R  C:\DATA\SALES89.WK1
```

The letter *R* that appears next to the files INCOME89.WK1 and SALES89.WK1 tells you that those files are read-only, and therefore can't be modified or deleted. In DOS jargon, a file that you can't modify has its read-only attribute turned on, while a file that you can modify has its read-only attribute turned off.

As you might guess, you can view the read-only status of a particular file or group of files by simply following the word *attrib* with the appropriate filename or filename descriptor. For instance, to view the read-only status of the files whose extensions are EXE, use the command.

```
C:\>attrib *.exe
```

Immediately, DOS will display the status of the current directory's EXE files.

Making a file read-only

To make a file read-only—that is, to protect a file against any attempts to change it or delete it—issue the command

```
attrib +r filename
```

where *filename* is the name of the file you want to protect.

Once you've used the ATTRIB command to turn on a file's read-only attribute, attempting to remove that file from your disk using the DEL or ERASE command will generate the message

```
Access denied
```

Continued on page 12

IN THIS ISSUE

A Publication of The Cobb Group

COBB

Inside DOS™

Inside DOS (ISSN 1049-5320) is published monthly by The Cobb Group, Inc., 9420 Bunsen Parkway, Suite 300, Louisville, KY 40220. Domestic subscriptions: 12 issues, $59. Foreign: 12 issues, $79. Single issues: $7 domestic, $8.50 foreign. Send subscriptions, fulfillment questions, and requests for bulk orders to Customer Relations, 9420 Bunsen Parkway, Suite 300, Louisville, KY 40220, or call (800)223-8720 or (502)491-1900. Our FAX number is (502)491-4200. Address correspondence and special requests to The Editor, *Inside DOS*, at the address above.

POSTMASTER: Send address changes to *Inside DOS*, P.O. Box 35160, Louisville, KY 40232. Second class postage is paid in Louisville, KY.

Editor-in-Chief:	Mark W. Crane
Technical Consultant:	Tim Landgrave
Contributing Editor:	Van Wolverton
Editing:	Jim Welp
	Toni Frank Bowers
Production:	Gina M. Sledge
	Julie Jefferson
Design:	Karl Feige
Publisher:	Douglas Cobb

STATEMENT OF OWNERSHIP, MANAGEMENT, AND CIRCULATION

Statement of ownership, management, and circulation (Required by 39 U.S.C. 3685). 1A. Title of publication: *Inside DOS*. 1B, Publication No. 10495320. 2. Date of filing: September 28, 1990. 3. Frequency of issue: Monthly. 3A. No. of issues published annually: 12. 3B Annual subscription price: $39 ($59 foreign). 4. Complete mailing address of known office of publication: The Cobb Group, Inc., 9420 Bunsen Parkway, Suite 300, Louisville, Kentucky 40220. 5. Complete mailing address of headquarters of general business offices of the publisher: The Cobb Group, Inc., 9420 Bunsen Parkway, Suite 300, Louisville, Kentucky 40220. 6. Full names and complete mailing address of publisher, editor, and managing editor: Publisher, Douglas Cobb, The Cobb Group, Inc., 9420 Bunsen Parkway, Suite 300, Louisville, Kentucky 40220; Editor, Mark Crane, The Cobb Group, Inc., 9420 Bunsen Parkway, Suite 300, Louisville, Kentucky 40220. Managing Editor, Kathy Chesky, The Cobb Group, Inc., 9420 Bunsen Parkway, Suite 300, Louisville, Kentucky 40220. 7. Owner: The Cobb Group, Inc., Douglas Cobb, Gena Cobb, Stewart Cobb, Steve Cobb, Ernestine H. Berg, Tom Cottingham, the address for all of the foregoing being 9420 Bunsen Parkway, Suite 300, Louisville, Kentucky 40220, and Barbara T. Wells, 1205 Elm Road, Louisville, Kentucky 40223. 8. Known bondholders, mortgagees, and other security holders owning or holding 1 percent or more of total amount of bonds, mortgages or other securities: None. 9. For completion by nonprofit organizations authorized to mail at special rates (DMM Section 423.12 only)—The purpose, function, and nonprofit status of this organization and the exempt status for Federal income tax purposes (Check one):__Has not changed during preceding 12 months__Has changed during preceding 12 months. (If changed, publisher must submit explanation of change with this statement.) 10. Extent and nature of circulation—A. Total no. copies (net press run): Average no. copies each issue during preceding 12 months, 12,794; actual no. copies of single issue published nearest to filing date, 20,206. B. Paid and/or requested circulation—1. Sales through dealers and carriers, street vendors and counter sales: Average no. copies each issue during preceding 12 months, 764; actual no. copies of single issue published nearest to filing date, 1,500. 2. Mail subscription (Paid and/or requested): Average no. copies each issue during preceding 12 months, 10,414; actual no. copies of single issue published nearest to filing date, 18,657. C. Total paid and/or requested circulation: (Sum of 10B1 and 10B2) Average no. copies each issue during preceding 12 months, 17,178; actual no. copies of single issue published nearest to filing date, 20,157. D. Free distribution by mail, carrier or other means. Samples, complimentary, and other free copies: Average no. copies each issue during preceding 12 months, 3; actual no. copies of single issue published nearest to filing date, 5. E. Total distribution (Sum of C and D): Average no. copies each issue during preceding 12 months, 11,181; actual no. copies of single issue published nearest to filing date, 20,162. F. Copies not distributed. 1. Office use, left over, unaccounted, spoiled after printing: Average no. copies each issue during preceding 12 months, 1,613; actual no. copies of single issue published nearest to filing date, 44. 2. Return from news agents. Average no. copies each issue during preceding 12 months, 0; actual no. copies of single issue published nearest to filing date, 0. G. Total (Sum of E, F1 and 2—should equal net press run shown in A): Average no. copies each issue during preceding 12 months, 12,794; actual no. copies of single issue published nearest to filing date, 20,206. 11. I certify that the statements made by me above are correct and complete. Douglas F. Cobb, Publisher.

Listing files by name or by size

Here's a simple tip I use all the time. As you know, the DIR command doesn't list files in alphabetical order. As a result, it's a real pain to issue the DIR command, and then to scan the directory listing carefully to find the particular file (or directory) you're looking for.

To overcome this problem, I use the command

```
C>dir | sort
```

to view the directory listing in alphabetical order. That way, I can quickly scan down the listing and identify the desired entry.

When I want to print a directory listing in alphabetical order, I use the command

```
C>dir | sort > prn
```

I find these two commands extremely helpful—I hope your readers do too.

Frances Henderson
Addison, Illinois

Ms. Henderson's techniques are handy for generating listings from A to Z. If you want, however, you can use a similar form of the DIR command in conjunction with the SORT filter to list entries from Z to A. To do this, issue the command

```
C>dir | sort /r
```

In addition to listing files alphabetically, you may sometimes want to list files by size. This is particularly useful when you need to delete some files from your hard disk to make some room. To list files by size from smallest to largest, use the command

```
C>dir | sort /+16
```

To list them from largest to smallest, use the command

```
C>dir | sort /r /+16
```

As Ms. Henderson pointed out, you can send a directory listing to the printer any time you want by appending the suffix > *prn* to the command. Additionally, you can view the listing one screen at a time by appending the suffix | *more*. For example, to view an alphabetical listing of files one screen at a time, use the command

```
C>dir | sort | more
```

Incidentally, all of these commands use piping (|) and/or redirection (>). In a future issue of *Inside DOS*, we'll explain piping and redirection in detail.

Replacing your computer's batteries

Did you know that your computer uses batteries? Ever since IBM introduced the PC-AT back in 1986, most personal computers have used batteries for two important tasks:

1) to keep the computer's internal system clock running when the computer is turned off
2) to power the memory that stores the computer's configuration settings, such as the type of disk drives installed, the amount of base memory and extended memory, and the type of display

If you're not sure whether your computer uses batteries, here's an easy way to make the determination: Simply turn your computer off and back on again. If the computer's date and time reset to a particular date when you do this, the computer probably doesn't use batteries; if the date and time stay current, however, the computer uses batteries.

Of course, your computer's batteries will eventually run down and need to be replaced. When this happens, the computer will lose track of the date and time as well as its configuration settings the next time you turn off the computer.

Depending on how often you power down your machine, most computer batteries last at least three years. Consequently, if your system is over three years old, you should buy a new set of batteries and replace the olds ones. Here are the basic steps necessary to replace your computer's batteries:

- record your computer's configuration settings
- remove the cover and replace the batteries
- turn on your computer and restore the configuration settings

Let's discuss each of these steps in a little more detail. By the way, if you use an IBM PS/2 computer, the technique you'll use is a little different from that for other PCs—more on that in a moment.

Recording the configuration settings

Before you remove the computer's old batteries (or before they run down), you need to bring up and record the computer's configuration settings. The method you use to bring up your computer's configuration settings depends on the brand and model of the computer you're using. As we noted earlier, some computers don't even use batteries—instead, they have a set of DIP switches on the motherboard that specify the computer's configuration.

If you're using an IBM PC-AT, you'll specify configuration settings by inserting the Setup and Diagnostics disk into drive A and rebooting your machine (the Setup and Diagnostics disk is stored in a plastic sleeve near the back of your computer's setup documentation). When the startup screen appears, choose Run Setup to bring up the computer's configuration settings.

If you're using an IBM-compatible computer (like Dell, Northgate, or Zeos), you'll bring up the computer's configuration screen either by using the computer's configuration disk, or by pressing a key sequence like [Ctrl][Alt][Enter] or [Ctrl][Alt][Esc].

Once you've figured out how to bring up your computer's configuration settings, write them down—all of them. You'll see all kinds of settings for the date and time, diskette drives, hard disks, base memory, extended memory, display type, and so forth. Make absolutely certain you record all the detailed information regarding your hard drive—if your computer permanently loses this information, you may not be able to access that drive. Even worse, if you later key in the wrong information for your hard drive, you can permanently damage that drive.

Removing the computer's cover and replacing the batteries

Replacing the batteries in your computer isn't as easy as replacing the batteries in your flashlight, but it's still pretty easy. Your computer owner's manual should offer step-by-step instructions (often accompanied by lots of neat, little sketches) for replacing the computer's batteries. Essentially, all you have to do is:

- turn off your computer
- remove the screws that hold the computer's cover in place
- remove the old batteries
- install the new batteries
- replace the computer's cover and re-insert the screws

If you've never removed the cover of your computer before, you'll probably want to enlist the help of a PC support specialist. The entire procedure should require no more than fifteen minutes.

Of course, your manual should tell you what kind of batteries your computer uses. If you can't find your owner's manual, or if the manual doesn't specify the type of batteries required, take your existing batteries to a local computer store and ask someone to help you find the appropriate replacements.

Restoring the configuration settings

Once you have installed new batteries and have replaced the computer's cover, you'll need to restore the computer's configuration settings, which were lost as soon as you removed the old batteries. To restore those settings, turn on your computer and use the same method you used before to access your computer's configuration screen. Once that screen appears, type in all the information exactly as you've recorded it. When you're finished, press the appropriate key to save your new settings and reboot the computer. At that point, your computer should boot up and once again operate normally.

For IBM PS/2 users

Setting up the configuration for an IBM PS/2 computer is somewhat different from setting up most other computers. If you're using an IBM PS/2, you won't need to write down your computer's configuration settings before you change its batteries. Instead, you can replace the batteries, and then let the computer's Reference Disk do the rest of the work for you. Simply insert the computer's Reference Disk into drive A and reboot the machine. When you see the message *Automatically configure the system? (Y/N)*, simply press *y*. The Reference Disk is nearly always smart enough to figure out what kind of equipment is installed on the computer, and can automatically set up the proper configuration.

However, if your computer is equipped with some special peripherals and device drivers, the Reference Disk may not be able to configure your system automatically. Instead, you'll need to follow the instructions in your owner's manual for configuring your machine. For more information on configuring an IBM PS/2, consult your local PC support specialist.

COPYING TECHNIQUE

Copying all files except those you specify

As you know, the COPY and XCOPY commands allow you to copy to another disk or directory only the group of files you specify. For instance, you can copy all the files in the current directory whose extensions are DBF to drive A using either of the following commands:

```
copy *.dbf a:
xcopy *.dbf a:
```

However, what can you do when you want to copy all the files in the current directory *except* those with a particular extension? For instance, suppose the current directory contains a group of very large files whose extensions are DBF, and you want to copy to another drive all the files except those with DBF extensions.

In this article, we'll explain a technique that uses simple DOS commands to copy all the files except those you specify. Then, we'll show you how to create a batch file that lets you perform this type of copy with a single command.

The technique

Our technique for copying all the files in the current directory except those you specify is as follows: First, you turn on the archive attribute for all the files in the current directory using the command:

```
attrib +a *.*
```

Next, you turn off the archive attribute for those files you do not want to copy using the command

```
attrib -a fileDescriptor
```

where *fileDescriptor* is the group of characters and wildcards (such as *.DBF) that specify those files. Finally, you copy those files whose archive attribute is turned on using the command

```
xcopy *.* destination /m
```

where *destination* is the destination drive and/or directory. Let's demonstrate this technique with a simple example.

An example

Suppose you want to copy to drive A all the files in the directory C:\MYPROG except those with DBF filename extensions. To do this, move into the C:\MYPROG directory, and turn on the archive attribute for all the files in that directory using the command

```
C:\MYPROG>attrib +a *.*
```

Next, turn off the archive attribute of the files whose extensions are DBF using the command

```
C:\MYPROG>attrib -a *.dbf
```

Finally, copy to drive A all the files whose archive attribute is still on using the command

```
C:\MYPROG>xcopy *.* a: /m
```

As soon as you do this, DOS will begin copying files from the source directory to drive A.

If all the files you specify will fit onto drive A, you'll see the DOS prompt again as soon as the last file is copied. However, if the disk in drive A can't hold all of the source files, DOS will copy as many files as it can, then present the message

```
Insufficient disk space
```

and return you to the prompt. However, because you've used the */m* option in the XCOPY command (which tells DOS to turn off each file's attribute as soon as it copies the file) you can continue copying the rest of the files by replacing the diskette and reissuing the command

```
C:\MYPROG>xcopy *.* a: /m
```

Continue replacing diskettes and reissuing this command until DOS finishes copying all the source files. At that point, if you check the target diskettes, you'll see that DOS has copied all the files in the directory C:\MYPROG except those with DBF extensions.

Placing the commands in NOCOPY.BAT

Although the copying technique we've explained works well, you'll want to make things easy for yourself by using the batch file NOCOPY.BAT, which is shown in Figure A. Once you've created this batch file and copied it into a directory whose name is on the path, here's how you use it to perform the copy: Simply move into the directory that contains the source files, then issue the command

```
nocopy fileDescriptor destination
```

where *fileDescriptor* is the group of characters and wildcards that describe the files you don't want to copy, and *destination* is the destination drive and/or directory. Let's see how this batch file works line by line.

Figure A

```
@echo off
attrib +a *.*
attrib -a %1
xcopy *.* %2 /m
```

NOCOPY.BAT lets you copy all the files except those you specify.

As you probably know, the first command in this batch file

```
@echo off
```

tells DOS not to display the batch file's commands as it executes them. The second command in this batch file

```
attrib +a *.*
```

tells DOS to turn on the archive attribute for every file in the current directory. The next command

```
attrib -a %1
```

uses the replaceable parameter *%1* to collect the *fileDescriptor* argument you supplied on the command line, and tells DOS to turn off the archive attribute for the files specified by this argument. The last command in the batch file

```
xcopy *.* %2 /m
```

uses the replaceable parameter *%2* to collect the *destination* argument you typed on the command line, and tells DOS to copy to that destination all the files whose archive attribute is turned on.

An example

To demonstrate how the NOCOPY.BAT file works, let's use this batch file to perform the same task we performed earlier. That is, let's copy to drive A all the files in the directory C:\MYPROG except those with DBF filename extensions. To do this, move into the directory C:\MYPROG, and issue the command

```
C:\MYPROG>nocopy *.dbf a:
```

As soon as you issue this command, DOS will begin copying the source files to drive A.

If the diskette in drive A isn't large enough to hold all the source files you've specified, you can't copy the rest of the files by simply rerunning NOCOPY.BAT. Instead, you'll need to replace the diskette in drive A and issue the command

```
C:\MYPROG>xcopy *.* a: /m
```

as we did before. Continue swapping diskets and reissuing this command until DOS successfully copies all the source files to drive A.

Conclusion

In this article, we've shown you a technique for copying all the files in a directory *except* for those you specify. Although our examples showed you how to use our technique to copy files to drive A, you can use this technique to copy files to any destination drive or directory.

VAN WOLVERTON

This article was written by Contributing Editor Van Wolverton. Van is the author of the bestselling books Running MS-DOS *and* Supercharging MS-DOS.

Getting to know a handy batch file tool: the IF command

As ad writers love to tell us, the uses of a computer are limited only by our own imagination. We can write novels or marketing proposals, plan budgets or keep fantasy football statistics, design houses or lay out weaving patterns. But it takes more than imagination, of course: We've got to learn how to use the programs that make a computer something more than a very expensive paperweight.

The better we learn to use application programs—word processors, for example, and spreadsheets and graphics programs—the more uses we find for these marvelous machines. Even if we buy a computer for one specific purpose, learning and using an application program almost always leads us to discover new uses. We need knowledge and experience to liberate our imaginations; the ad writers don't tell us that.

But somehow, in the rush to learn how to use programs, there's one program that everyone uses but not everyone learns how to use. It's DOS, of course, and like any other program, it requires some effort to learn. It's all too common an occurence for people to buy a computer, learn to use one or two application programs, but put off learning anything about DOS. It won't take long for them to encounter a situation they can't handle.

Item: An amateur historian learns to use a database program to record the names of all the Civil War veterans from a western state. One day, after months of data entry, the program refuses to index the database, reporting that the disk is full. At a friend's suggestion, our amateur historian buys another fixed disk but, without knowing how to use DOS, can't transfer the program or data to the new (larger) disk.

Item: A parent buys a computer so the children can use it for school work, and learns to use Microsoft Works for word processing and household budgeting. Seeing an ad for some shareware programs, the parent buys several disks of software that really sound interesting, but can't figure out how to get the programs on the fixed disk.

Someone who uses a computer without at least a rudimentary knowledge of disks, files, and basic DOS commands is flying blind. Sooner or later it will catch up with them.

Tailoring with batch files

Knowing how to use DOS not only lets you manage your system more effectively, it gives you access to one of the handiest features of DOS: batch files. As you probably know, batch files let you personalize your system by tailoring the startup procedure to your devices and programs, and by automating commands that you use frequently.

Learning to write batch files takes some study and practice, just like learning to use any other program. Magazines often include articles about batch files; entire books, in fact, have been written about them. Because of their importance, each issue of *Inside DOS* includes tips and techniques for writing batch files. If you haven't yet written any batch files, take the time to hunt up some of the introductory articles and jump in.

The IF batch command

DOS includes a few commands specifically designed for use in batch files. One of the most useful is the IF command, which lets you control how the batch file operates depending on a certain condition. The IF command lets you check three conditions:

- whether a file exists
- the value of the variable ERRORLEVEL
- whether two series of characters (strings) are the same

A few examples show how useful the IF command can be.

Does that file exist?

Checking whether a file exists is a valuable tool. For example, Copy appears to be a fairly benign DOS command, but it can be one of the most dangerous because it lets you erase a file without even knowing it. If you copy a file and tell DOS to give the copy (the *target* filename) the same name as an existing file, DOS dutifully replaces the existing file with the new copy and simply responds *1 File(s) copied*. You haven't got a clue that the existing file is gone; when you do discover it, you can't even recover it with a file-recovery program. It's gone forever.

You can protect yourself against that sort of pilot error by writing a batch file that won't copy the file if the target file already exists. Such a batch file (we'll call it SAFECOPY.BAT) would take two parameters, just like the Copy command: The first (referred to as %1 in the batch file) names the source file and the second (%2) names the target file. SAFECOPY.BAT requires a single IF command that copies the source file only if the target file *doesn't* exist:

```
if not exist %2 copy %1 %2
```

To make SAFECOPY.BAT more useful, however, you'd want it to display a message if the target file exists and the source file isn't copied. Figure A shows an improved version of SAFECOPY.BAT that does the job nicely.

Figure A

```
@echo off
if exist %2 goto NOTSAFE
copy %1 %2
goto end
:NOTSAFE
echo File %2 exists and
echo would be replaced by %1.
echo FILE NOT COPIED.
:END
```

SAFECOPY.BAT protects you against accidentally copying over an existing file.

To use this batch file to copy FEBSALES.XLS to FEBSALES.SAV, you would type

```
C>safecopy febsales.xls febsales.sav
```

You can use wildcard characters with SAFECOPY.BAT, but wildcard characters in the target filename restrict its use. For example, suppose you made a copy of FEBSALES.XLS called FEBSALES.SAV as in the previous example, then decided to copy all files that begin with FEB and whose extensions are XLS to files with the same name whose extensions are SAV. You would type

```
C>safecopy feb*.xls *.sav
```

but no files would be copied because there is already a file whose extension is SAV.

Admittedly, SAFECOPY.BAT isn't quite as flexible as the COPY command—but it's a lot safer.

What's an ERRORLEVEL?

DOS keeps track of a numeric value called ERRORLEVEL, which any program can set. DOS lets you check ERRORLEVEL with the IF command, giving programs a way to communicate with you. This is of pretty limited value, because only a few commands (such as BACKUP and RESTORE) set ERRORLEVEL.

But as the article on page 8 of this issue shows, it isn't difficult to create a program that sets ERRORLEVEL to a value that identifies the last key that was pressed. This makes ERRORLEVEL more useful, because now you can write interactive batch files that change their operation depending on which key is pressed. For more information on this technique, read the article on page 8, or refer to the article entitled "Soliciting Input in a Batch File," which appeared in the September issue.

Comparing Strings

The last option of the IF command lets you check whether one string of characters is the same as another. One of the most common uses of this is to see what parameters, if any, were typed with a batch command. This is pretty straightforward. For example, suppose you wanted a batch file to go to the label END if the first parameter typed with the batch command was off, you'd use the following IF command:

```
if %1==off goto end
```

If the first parameter is anything other than *off*, DOS ignores the GOTO command and continues with whatever commands follow the IF command.

But there's an aesthetic problem: If no parameter was typed with the batch command, DOS displays *Syntax error*, then continues with the commands that follow the IF command. The message is not only unattractive, it can be unsettling.

There's also a more serious problem: What if you want the batch file to do something only if no parameter is typed (or if one parameter is typed, but not two)? How do you check for the absence of a parameter?

It's fairly simple. Just put quotation marks around the replaceable parameter and the string you want to compare it within the IF command. If you want to check for the absence of a parameter (this is sometimes called a "null" parameter), compare the replaceable parameter to two quotation marks with nothing between them.

For example, the following IF command checks whether the first parameter is *off* and avoids the error message if there is no parameter:

```
if "%1"=="off" goto end
```

It's just as easy to go to the label END if there is no parameter at all:

```
if "%1"=="" goto end
```

As you gain experience with batch files, you'll find that they can significantly improve the usability of your system, and the IF command is one of the most valuable tools in your batch file kit.

A better way to solicit input in a batch file

The idea for this article and its accompanying Debug script was submitted by John C. Bowers of Alta Loma, California.

In a previous issue of *Inside DOS*, we showed you how to use Debug to create a program called KEYPRESS.COM, and then explained how to use that utility to solicit a *yes* or *no* response in a batch file ("Soliciting Input in a Batch File," September 1990). Although this technique works well when you want to ask yes or no questions, you'll sometimes want to ask questions that would require the user to press a key other than Y or N. For instance, you might want to ask a question that offers three or more possible responses, or you might want to use a letter that is more appropriate than Y or N (as in, "Press Q to Quit").

In this article, we'll present a more sophisticated version of KEYPRESS.COM that overcomes some of the limitations inherent in the original version. For instance, our new KEYPRESS.COM will allow you to solicit for any alphabetic key (not just Y, N, or ESC). It also lets you specify the prompting message as an argument to the KEYPRESS command. After we show you how to create the new KEYPRESS.COM using Debug, we'll demonstrate how it's used in a couple of sample batch files.

Creating the new KEYPRESS.COM

To create the new version of KEYPRESS.COM, you'll need to use the Debug script shown in Figure A. As you may recall from September, you should use a text editor (like Edlin) to create a text file that contains these Debug instructions. If you don't know how to use a text editor to create the Debug script in Figure A, refer to the September issue, which provides step-by-step instructions for using Edlin to create such a file.

By the way, all of the *0*s in Figure A are zeros—not letter *O*s. Additionally, all the *1*s are number *1*s—not lowercase letter *L*s. Finally, make sure you include a blank line after the letter *q* at the very bottom of the file.

Once you've created the Debug script shown in Figure A, you're ready to run Debug and create the file KEYPRESS.COM. Assuming you've named your script file KEYPRESS.SCR, you'll want to issue the command

```
C>debug < keypress.scr
```

Immediately, Debug will execute the instructions in the file KEYPRESS.SCR, and will deposit into the current directory the file KEYPRESS.COM.

Once you've created KEYPRESS.COM, copy it into a directory whose name is on the path. That way, DOS will always be able to find and execute this file when you call it from within a batch file.

Figure A

```
n keypress.com
e 100 BA 82 00 8B FA 8A 4D FE
e 108 8B F1 4E FE C9 7F 05 2B
e 110 C0 EB 56 90 8A 9C 81 00
e 118 80 FB 5D 74 09 4E 7F F4
e 120 E8 77 00 EB 44 90 8B FE
e 128 4F 4E 74 0E 8A 9C 81 00
e 130 80 FB 5B 75 F4 4E 7E 05
e 138 8B CE E8 5D 00 46 46 8B
e 140 CE E8 4F 00 E8 5B 00 73
e 148 05 E8 3E 00 EB F3 8A C3
e 150 8A 9C 81 00 E8 4B 00 72
e 158 04 3A C3 74 29 46 3B F7
e 160 7E EE 8B F1 E8 23 00 EB
e 168 D8 E8 27 00 E8 33 00 8A
e 170 C3 3C 19 75 02 EB 0F 3C
e 178 0E 75 02 EB 09 3C 1B 74
e 180 05 E8 06 00 EB E3 B4 4C
e 188 CD 21 52 B2 07 B4 02 CD
e 190 21 5A C3 B4 08 CD 21 8A
e 198 D8 C3 B4 40 BB 01 00 CD
e 1A0 21 C3 80 FB 1B 74 1C 80
e 1A8 FB 61 7C 0A 80 FB 7A 7F
e 1B0 14 80 EB 20 EB 0A 80 FB
e 1B8 41 7C 0A 80 FB 5A 7F 05
e 1C0 80 EB 40 F8 C3 F9 C3
rcx
C7
w
q
(blank line)
```

You can use this Debug script to create the new and improved KEYPRESS.COM.

Using KEYPRESS.COM in a batch file

Unlike the old version of KEYPRESS.COM, which accepts no arguments, the new and improved KEYPRESS.COM lets you specify both the prompting message as well as the list of valid keys the user can press. The new command takes the form

```
keypress prompt [validKeys]
```

where *prompt* is the prompting message you want to display on the screen, and *validKeys* is the list of valid keys the user can press. For instance, if you want to display the prompting message *Press Y or N:* and accept only those two keys as a valid response, you would use the command

```
keypress Press Y or N: [yn]
```

As soon as the user presses a valid key, KEYPRESS.COM sets the value of the system variable ERRORLEVEL to the corresponding value shown in Table A. For instance, if the user presses Y (or y), KEYPRESS.COM will set ERRORLEVEL equal to 25.

Once you've used KEYPRESS.COM to collect your input, you can use a series of IF commands to check for values of ERRORLEVEL and branch to the appropriate section of the batch file. Keep in mind, however, that you need to check for values of ERRORLEVEL in *descending* order. That is, you need to check if ERRORLEVEL is equal to higher numbers first, and lower numbers second. If you check for the values of ERRORLEVEL in ascending rather than descending order, your IF statements won't allow DOS to branch properly.

Table A

A	1	J	10	S	19
B	2	K	11	T	20
C	3	L	12	U	21
D	4	M	13	V	22
E	5	N	14	W	23
F	6	O	15	X	24
G	7	P	16	Y	25
H	8	Q	17	Z	26
I	9	R	18	ESC	27

KEYPRESS.COM sets ERRORLEVEL to the value associated with the key you press.

An example

As an example, suppose you want to prompt the user to press A, B, or C, and then execute a specific sequence of commands depending on which key they press. To do this, you could use a batch file like the one shown in Figure B.

The first command in KEYTEST1.BAT

```
@echo off
```

simply tells DOS not to display the batch file's commands as it executes them. (By the way, if you're using version 3.2 or earlier of DOS, you'll need to omit the @ sign in front of the *echo off* command.) The next command

```
keypress Press A, B, or C:[abc]
```

runs KEYPRESS.COM, which tells DOS to display the message

```
Press A, B, or C:_
```

and to wait until the user presses A, B, or C (or a, b, or c). Because we supplied the argument [abc] to our KEYPRESS command, DOS will not execute the next command in the batch file until you press one of these three valid keys.

Figure B

```
@echo off
keypress Press A, B, or C: [abc]
if errorlevel 3 goto C
if errorlevel 2 goto B
if errorlevel 1 goto A
:C
echo:
echo You selected C
goto END
:B
echo:
echo You selected B
goto END
:A
echo:
echo You selected A
:END
```

KEYTEST1.BAT prompts the user to press A, B, or C, and then executes the appropriate sequence of commands.

The remainder of the batch file takes the same form as the batch file we presented in our previous article on KEYPRESS.COM. The next three commands in KEYTEST1.BAT

```
if errorlevel 3 goto C
if errorlevel 2 goto B
if errorlevel 1 goto A
```

check to see if ERRORLEVEL is equal to 3, 2, or 1, respectively, and if so, they tell DOS to branch to the appropriate section of the batch file (the section that begins with the label *:C*, *:B*, or *:A*). Notice that we've followed the rule of checking for values of ERRORLEVEL in descending order.

Of course, the sections labelled *:C* and *:B* both end with the command

```
goto end
```

which tells DOS to skip directly to the last line in the batch file, thus ending the batch file's execution.

When you run KEYTEST1.BAT by typing

```
C>keytest1
```

and pressing ↵, DOS will display the message

```
Press A, B, or C:_
```

If your press A (or a), you'll then see the message

```
You selected A
```

Of course, if you run the batch file again and press either B or C, DOS will tell you that you selected that key instead.

Another example

Now let's suppose you want to prompt the user to press S (for Start), B (for Backup), or E (for Exit), and then execute the appropriate sequence of commands. Figure C shows a batch file that will do the job.

Figure C

```
@echo off
keypress Press S (for Start), B (for Back
   up), or E (for Exit): [sbe]
if errorlevel 19 goto S
if errorlevel 5 goto E
if errorlevel 2 goto B
:S
echo:
echo You selected Start
goto end
:E
echo:
echo You selected Exit
goto end
:B
echo:
echo You selected Backup
:END
```

KEYTEST2.BAT prompts the user to press S, B, or E, and then executes the appropriate sequence of commands.

As you can see, this batch file has the same basic structure as the previous batch file. The only difference is that here we've used the command

```
keypress Press S (for Start), B (for Back
   up), or E (for Exit): [sbe]
```

to solicit for the letters S, B, or E. If you run this batch file and press any key other than S, B, or E (or s, b, or e), DOS will simply beep and wait for you to press a valid key before it carries out any additional commands.

Other forms of the KEYPRESS command

If you want, you can omit the prompt argument from the KEYPRESS command and use an ECHO command to display the prompting message. For instance, if you want to solicit for the keys A, B, or C, you can use either the command

```
keypress Press A, B, or C: [abc]
```

or the commands

```
echo Press A, B, or C
keypress [abc]
```

You'll typically want to include the *prompt* argument in the KEYPRESS command, though, since DOS can execute a single KEYPRESS command faster than an ECHO and KEYPRESS command pair.

Additionally, if you want to accept any keys as valid options, you can omit the *validKeys* argument in your KEYPRESS command. In fact, if you want to accept the ESC key as a valid option, you must omit the *validKeys* argument. For instance, to solicit for the keys Y, N, or ESC, you'll use the command

```
keypress Press Y, N, or ESC:
```

then follow that command with IF statements that check to see if ERRORLEVEL is equal to 27 (for ESC), to 25 (for Y), or to 14 (for N).

Conclusion

Although our September issue presented a useful utility program called KEYPRESS.COM, that version of KEYPRESS.COM had a couple of limitations. In this article, we've presented a more sophisticated version of KEYPRESS.COM that overcomes a couple of those limitations, and we've demonstrated how to use this utility to solicit input in a batch file.

Displaying a blank line in a batch file

When you write a batch file that displays text on the screen, you'll usually want to insert a blank line to make the text more attractive and easier to read. For instance, if you create a menu system for your computer, you'll want to insert a blank line between the menu headings and the options on each menu. In this article, we'll show you how to use the ECHO command to create a blank line in your batch files.

The problem

As you probably know, the ECHO command displays text on the screen. For example, the command

```
echo Good Morning!
```

displays the line

```
Good Morning!
```

As you may have discovered, however, you can't echo a blank line to the screen by simply typing the command

```
echo
```

If you do this, DOS displays either the message *ECHO is on* or the message *ECHO is off* instead of displaying a blank line.

The solution

Fortunately, you can use the ECHO command to display a blank line by echoing one of DOS' nonprinting characters to the screen. All you have to do is immediately follow the word *echo* with the appropriate character. For versions 3.x or 4.x, all of the following commands will produce a blank line:

```
echo.
echo+
echo/
echo[
echo]
echo:
```

For earlier DOS versions, you'll need to follow the ECHO command with one of these characters:

tab
comma
line feed
space
semicolon
equal sign

as well as a trailing space. For instance, under DOS 2.1 you can produce a blank line by typing *echo*, immediately followed by a tab and a space. (Notice that later versions of DOS don't care whether you follow the nonprinting character with a space, while earlier versions require the trailing space.)

If you want to write a batch file that generates a blank line in nearly every version of DOS, try typing the word *echo*, pressing the [F7] key, then typing a space. Pressing the [F7] key produces an @ sign in your batch file, but DOS interprets this character as a nonprinting character and therefore generates a blank line.

An example

Figure A shows a sample batch file that displays two lines of text separated by a blank line.

Figure A

```
@echo off
cls
echo Data Entry Menu
echo]
echo Enter your selection
```

This batch file produces a heading, a blank line, and a line of text.

When you run the batch file in Figure A, DOS will display the text

```
Data Entry Menu

Enter your selection
```

Printing multiple blank lines

You might think you could produce several blank lines in succession by following the word *echo* with several nonprinting characters. Unfortunately, this technique doesn't work. For instance, if you include in a batch file the command

```
echo]]]]
```

and then run that batch file, DOS will display the characters

```
]]]
```

instead of a blank line. For each blank line, then, you need to use a separate ECHO command that echoes a single nonprinting character.

THE COBB GROUP

Postmaster:
Monthly Technical Journal
Please Rush!
Second Class Postage

Please include account number from label with any correspondence.

Protecting a file against changes using ATTRIB

Continued from page 1

If you attempt to edit the file's contents using EDLIN, you'll see the message

```
File is READ-ONLY
```

Finally, if you attempt to edit a read-only file's contents using an application, you'll receive some kind of error message when you attempt to save changes to the file from within the application.

For example, suppose you want to make your AUTOEXEC.BAT file read-only so that no one (including you) can accidentally delete it. To do this, first move into the root directory and issue the command

```
C:\>attrib *.*
```

Immediately, DOS will show you the status of the files in the root directory

```
  A         C:\COMMAND.COM
  A         C:\AUTOEXEC.BAT
  A         C:\CONFIG.SYS
```

As you can see, the letter *R* does not appear next to the file AUTOEXEC.BAT, indicating that its read-only attribute is turned off.

Now, turn on AUTOEXEC.BAT's read-only attribute using the command

```
C:\>attrib +r autoexec.bat
```

At this point, if you issue the command

```
C:\>attrib *.*
```

DOS will display the letter *R* next to the file AUTOEXEC.BAT to indicate that the file is now read-only. In other words, no one will be able to modify your AUTOEXEC.BAT file unless they first use the ATTRIB command to turn off that file's read-only attribute.

Removing read-only protection

To remove a file's read-only status, (in other words, to turn off its read-only attribute), issue the command

```
attrib -r filename
```

where *filename* is the name of the file you want to unprotect. Once you've turned off a file's read-only attribute, of course, you won't see the letter *R* next to its name when you use the ATTRIB command to display the file's read-only status.

For example, to turn off the AUTOEXEC.BAT file's read-only attribute, issue the command

```
C:\>attrib -r autoexec.bat
```

Once you've issued this command, DOS will allow you to modify or delete AUTOEXEC.BAT, and will remove the letter *R* next to its name when you list it using the ATTRIB command.

By the way, if you explore the directories on your hard drive, you might discover that some of your application program files are read-only. If you attempt to remove an application from your hard disk by deleting the entire directory in which it's installed, you're likely to find that DOS won't delete some of the files in that directory until you use the ATTRIB command to turn off those files' read-only attributes. Anytime you attempt to delete a file and see the message *Access denied*, you'll know to turn off that file's read-only attribute using ATTRIB.

Notes

If you plan to use ATTRIB to protect some of your files, you'll want to remember a couple of important points. First of all, ATTRIB isn't a great security feature—if someone really wants to delete a file you've made read-only, all they have to do is use the ATTRIB command to turn off its read-only attribute. Therefore, ATTRIB is most useful for protecting against *accidental* deletions or modifications.

Additionally, if you make your AUTOEXEC.BAT and CONFIG.SYS files read-only, you might run into problems when you install new applications. Most applications offer a Setup or an Install program that automatically makes some changes to AUTOEXEC.BAT and CONFIG.SYS. For this reason, if you plan to make these two files read-only, remember to turn off their read-only attribute before you run a new application's Setup or Install program.